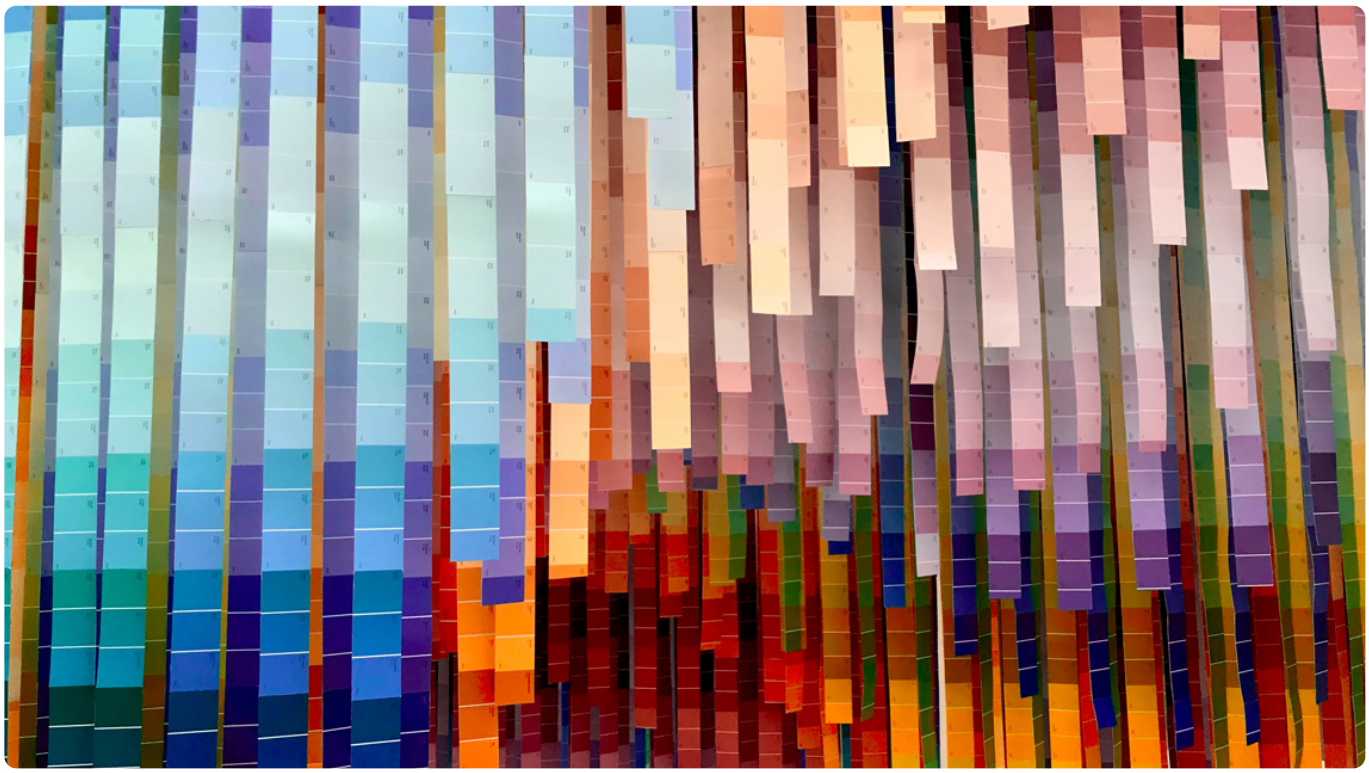
10 | 案例：在JMeter中如何设置参数化数据?

2020-01-06 高楼



讲述：高楼
时长 19:01 大小 17.42M

今天我们来做一个实例，看下在 JMeter 中，如何合理地设置参数化数据。

正式场景前的基准测试

在没有做业务混合场景之前，我们需要先做 Benchmark 测试，来确定一个登录业务能支持多少的业务量，这样就可以在业务混合场景中，根据场景中各业务的比例来确定登录的数据需要多少真实的数据。

我们继续用上一篇文章中用户登录的例子，单独测试登录业务，结果如下：

复制代码

```
1 Java
```

```
summary +    125 in 00:00:04 =   31.0/s Avg:    28 Min:     0 Max:   869 Err:
summary +   3404 in 00:00:30 =  113.2/s Avg:    31 Min:     0 Max:   361 Err:
summary +   4444 in 00:00:30 =  148.4/s Avg:    57 Min:     0 Max:   623 Err:
```

从上面的结果可以看到登录业务能达到的 TPS 是 113 左右，这里我们取整为 100，以方便后续的计算。

在测试工具中配置参数

在上面的试探性测试场景中，不需要观察系统的资源，只需要根据 TPS 做相应的数据统计即可。

前面我们知道，在这个示例中只做了近 10 万条的用户数据，为了方便示例进程。

下面我们从数据库中查询可以支持登录 5 分钟不重复的用户数据。根据前面的公式，我们需要 30000 条数据。

复制代码

```
Java
100x5mx60s=30000 条
```

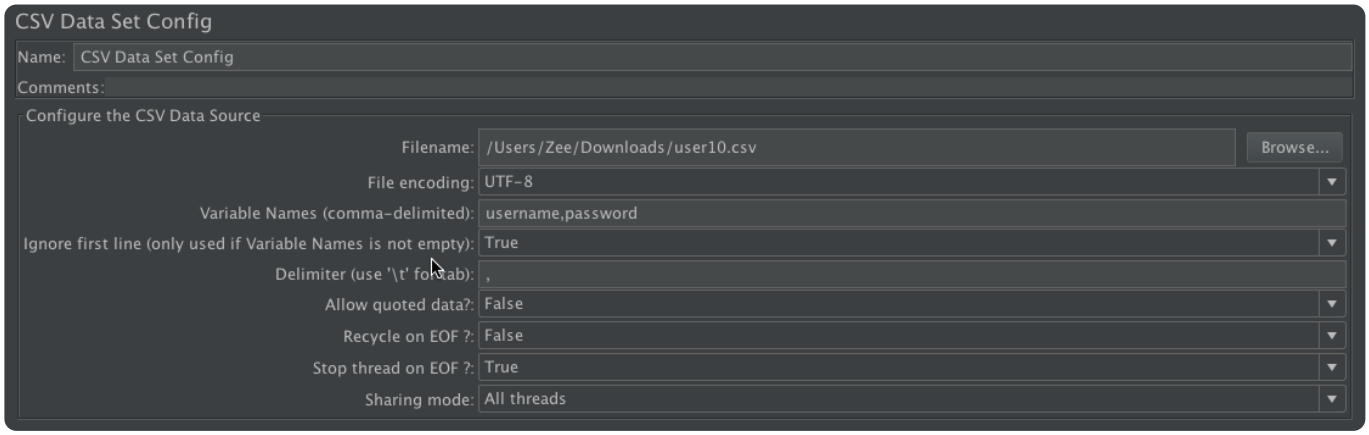
接下来连接数据库，取 30000 条数据，存放到文本中，如下所示：

复制代码

```
Java
username,password
test00001,test00001
test00002,test00002
test00003,test00003
test00004,test00004
test00005,test00005
test00006,test00006
test00007,test00007
.................
test30000,test30000
```

参数化配置在 JMeter 中的使用说明

我们将这些用户配置到测试工具的参数当中，这里以 JMeter 的 CSV Data Set Config 功能为例。配置如下：



在 JMeter 的参数化配置中，有几个技术点，在这里说明一下。

“Allow quoted data?”这里有两个选择，分别是 False 和 True。它的含义为是否允许带引号的数据，比如说在参数化文件中有这样的数据。

复制代码

```
Java
"username","password"
"test00001","test00001"
"test00002","test00002"
.....................
"test30000","test30000"
```

如果有引号，这个选择必须是 True。如果设置为 False，那么我们在脚本中会看到如下的数据：

复制代码

```
Java
username=%22test00001%22password=%22test00001%22
```

由于设置为 False，JMeter 将（"）转换为了 %22 的 URL 编码，很显然这个数据是错的。如果选择为 True，则显示如下：

```
Java
username=test00001password=test00001
```

这里就显示对了。

除此之外，还有如下几个功能点需要说明：

Recycle on EOF?：这里有三个选择，False、True 和 Edit。前两个选择非常容易理解。False 是指在没有参数的时候不循环使用；True 是指在没有参数的时候循环使用。Edit 是指在没有参数的时候会□根据定义的内容来调用函数或变量。

Stop thread on EOF?：这里有三个选择，False、True 和 Edit。含义和上面一致。

Sharing mode : 这里有四个选择，All threads、Current thread group、Current thread、Edit。

Sharing mode 的前三个选择是比较容易理解的，参数是在所有线程中生效，在当前线程组生效，还是在当前线程中生效。但这里的 Edit 和前两个参数中的 Edit 相比，有不同的含义。这里选择了 Edit 之后，会出现一个输入框，就是说这里并不是给引用函数和参数使用的，而是要自己□明确如何执行 Sharing mode。那如何来使用呢?

举例来说，假设我们有 Thread Group 1-5 四个线程组，但是参数化文件只想在 Thread Group 1、3、5 中使用，不想在线程组 2、4 中使用，那么很显然前面的几个选项都达不到目的，这时我们就可以选择 Edit 选项，在这里输入 `SharedWithThreadGroup1and3and5`。而在其他的线程组中配置其他参数化文件。

也就是说同样的一个变量名，在线程组 1/3/5 中取了一组数据，在线程组 2/4 中取了另一组数据。

以上三个参数的选项可以随意组合。于是就会得到如下表。

需要注意的是，EOF 是文件结束符的意思。在下面的解释中，为了更符合性能测试中的术语，特意解释为参数不足时。

以上三个功能点根据参数设计得不同，会产生不同的组合，我们依次查看一下。

序号	Recycle on EOF ?	Stop thread on EOF ?	Sharing mode	含义
1	False	False	All threads	1. 不可循环使用参数。 2. 在参数不足时不停止线程。 3. 在所有线程中生效。

这个组合显然是矛盾的，没有参数时不让循环，还不让停止线程，这不是耍流氓吗？真实的结果是什么呢？当我们执行时就会发现，参数变成了这样：

复制代码

```
username=%3CEOF%3E&password=%3CEOF%3E
```

服务端果然返回了：`{"flag":false,"errMsg":"账号不存在"}`。

序号	Recycle on EOF ?	Stop thread on EOF ?	Sharing mode	含义
2	True	False	All threads	1. 可循环使用参数。 2. 在参数不足时不停止线程。 3. 在所有线程中生效。

这个组合中第二个选项显然是没意义的，既然参数允许重复使用了，又怎么会发生参数不足停止线程的情况呢？

序号	Recycle on EOF ?	Stop thread on EOF ?	Sharing mode	含义
3	Edit	False	All threads	1. 参数不足时引用函数或变量。 2. 在参数不足时停止线程。 3. 在所有线程中生效。

这个组合因为第一个选项为“Edit”所以变得不确定了，如果在 Edit 的函数或变量返回为 True，则和第 2 种组合一样；如果返回为 False，则和第 1 种组合一样。

序号	Recycle on EOF ?	Stop thread on EOF ?	Sharing mode	含义
4	False	True	All threads	1. 不可循环使用参数。 2. 在参数不足时停止线程。 3. 在所有线程中生效。

这是一个完全合情合理的组合!

序号	Recycle on EOF ?	Stop thread on EOF ?	Sharing mode	含义
5	Ture	True	All threads	1. 可循环使用参数。 2. 在参数不足时停止线程。 3. 在所有线程中生效。

同第二个组合一样，第二个选项显然没有意义。

序号	Recycle on EOF ?	Stop thread on EOF ?	Sharing mode	含义
6	Edit	True	All threads	1. 参数不足时引用函数或变量。 2. 在参数不足时停止线程。 3. 在所有线程中生效。

这个组合同样因为第一个选项为 Edit，所以变得不确定了，如果在 Edit 的函数或变量返回为 True，则和第 3 种组合一样；如果返回为 False，则和第 4 种组合一样。

序号	Recycle on EOF ?	Stop thread on EOF ?	Sharing mode	含义
7	False	Edit	All threads	1. 不可循环使用参数。 2. 是否停止线程取决于edit的结果。 3. 在所有线程中生效。

这个组合因为是否停止线程的不确定性会出现两种可能，有可能是第 1 种组合，也有可能是第 4 种组合。

序号	Recycle on EOF ?	Stop thread on EOF ?	Sharing mode	含义
8	True	Edit	All threads	1. 可循环使用参数。 2. 是否停止线程取决于edit的结果。 3. 在所有线程中生效。

这个组合中是否停止线程的 Edit 配置没有意义，因为可循环使用参数，所以不会发生□参数不足导致线程停止的情况。

序号	Recycle on EOF ?	Stop thread on EOF ?	Sharing mode	含义
9	Edit	Edit	All threads	1. 参数不足时引用函数或变量。 2. 是否停止线程取决于edit的结果。 3. 在所有线程中生效。

这是一个古怪的组合，具有相当的不确定性，有可能变成第 1、2、4、5 种组合。

下面我们再来看下其他衍生的设置组合。

序号	Recycle on EOF ?	Stop thread on EOF ?	Sharing mode	含义
10	False	False	Current thread group	这个组合含义同第1种，只是生效范围在当前线程组中。
11	True	False	Current thread group	这个组合含义同第2种，只是生效范围在当前线程组中。
12	Edit	False	Current thread group	这个组合含义同第3种，只是生效范围在当前线程组中。
13	False	True	Current thread group	这个组合含义同第4种，只是生效范围在当前线程组中。
14	True	True	Current thread group	这个组合含义同第5种，只是生效范围在当前线程组中。

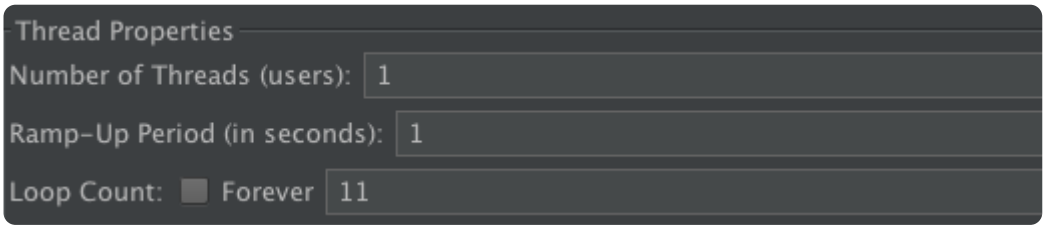
15	Edit	True	Current thread group	这个组合含义同第6种，只是生效范围在当前线程组中。
16	False	Edit	Current thread group	这个组合含义同第7种，只是生效范围在当前线程组中。
17	True	Edit	Current thread group	这个组合含义同第8种，只是生效范围在当前线程组中。
18	Edit	Edit	Current thread group	这个组合含义同第9种，只是生效范围在当前线程组中。
19	False	False	Current thread	这个组合含义同第1种，只是生效范围在当前线程中。
20	True	False	Current thread	这个组合含义同第2种，只是生效范围在当前线程中。
21	Edit	False	Current thread	这个组合含义同第3种，只是生效范围在当前线程中。
22	False	True	Current thread	这个组合含义同第4种，只是生效范围在当前线程中。
23	True	True	Current thread	这个组合含义同第5种，只是生效范围在当前线程中。
24	Edit	True	Current thread	这个组合含义同第6种，只是生效范围在当前线程中。
25	False	Edit	Current thread	这个组合含义同第7种，只是生效范围在当前线程中。
26	True	Edit	Current thread	这个组合含义同第8种，只是生效范围在当前线程中。
27	Edit	Edit	Current thread	这个组合含义同第9种，只是生效范围在当前线程中。
28	False	False	Edit	这个组合含义同第1种，只是生效范围取决于Edit的配置结果。
29	True	False	Edit	这个组合含义同第2种，只是生效范围取决于Edit的配置结果。
30	Edit	False	Edit	这个组合含义同第3种，只是生效范围取决于Edit的配置结果。
31	False	True	Edit	这个组合含义同第4种，只是生效范围取决于Edit的配置结果。
32	True	True	Edit	这个组合含义同第5种，只是生效范围取决于Edit的配置结果。
33	Edit	True	Edit	这个组合含义同第6种，只是生效范

				围取决于Edit的配置结果。
34	False	Edit	Edit	这个组合含义同第7种，只是生效范围取决于Edit的配置结果。
35	True	Edit	Edit	这个组合含义同第8种，只是生效范围取决于Edit的配置结果。
36	Edit	Edit	Edit	**这个组合太任性了！** 参数循环取决于设计的函数和变量的返回结果。 是否停止线程取决于函数或变量的返回结果。 作用域取决于自己的Edit配置结果。

真实场景下的 JMeter 参数配置和执行结果

根据以上的描述，我们先用 10 个用户来测试下，将 Stop `thread on EOF?`改为 True，将`Recycle on EOF?`改为 False，其他不变。同时将线程组中配置为 1 个线程循环 11 次。这样设置的目的是为了看在数据不足时，是否可以根据规则停掉线程组。如下所示：

线程组配置如下：



执行之后，我们会在日志中看到如下信息：

复制代码

```
1 Java
```

```
2 2019-09-05 22:56:30,171 INFO o.a.j.t.JMeterThread: Stop Thread seen for thread
```

可以看到在参数用完又不可循环使用参数的情况下，JMeter 主动停止了线程。

我们延续使用上文中场景二的条件，即希望场景中每个线程的每次迭代都用不同的数据。

为了能很快地整理出实际的结果，我们只使用 10 条数据来模拟，条件设置如下：

复制代码

```
1 线程组：2
2 线程（每线程组）：6
3 参数化数据：10 条
```

执行完场景后，会在日志中看到如下信息：

复制代码

```
1 Java
2 2019-09-07 23:24:25,585 INFO o.a.j.t.JMeterThread: Stop Thread seen for thread
3 2019-09-07 23:24:25,452 INFO o.a.j.t.JMeterThread: Stop Thread seen for thread
4 2019-09-07 23:24:23,406 INFO o.a.j.t.JMeterThread: Stop Thread seen for thread
5 2019-09-07 23:24:25,517 INFO o.a.j.t.JMeterThread: Stop Thread seen for thread
```

可见所有的线程都按我们的配置停止了线程，同时各线程取得参数如下表所示：

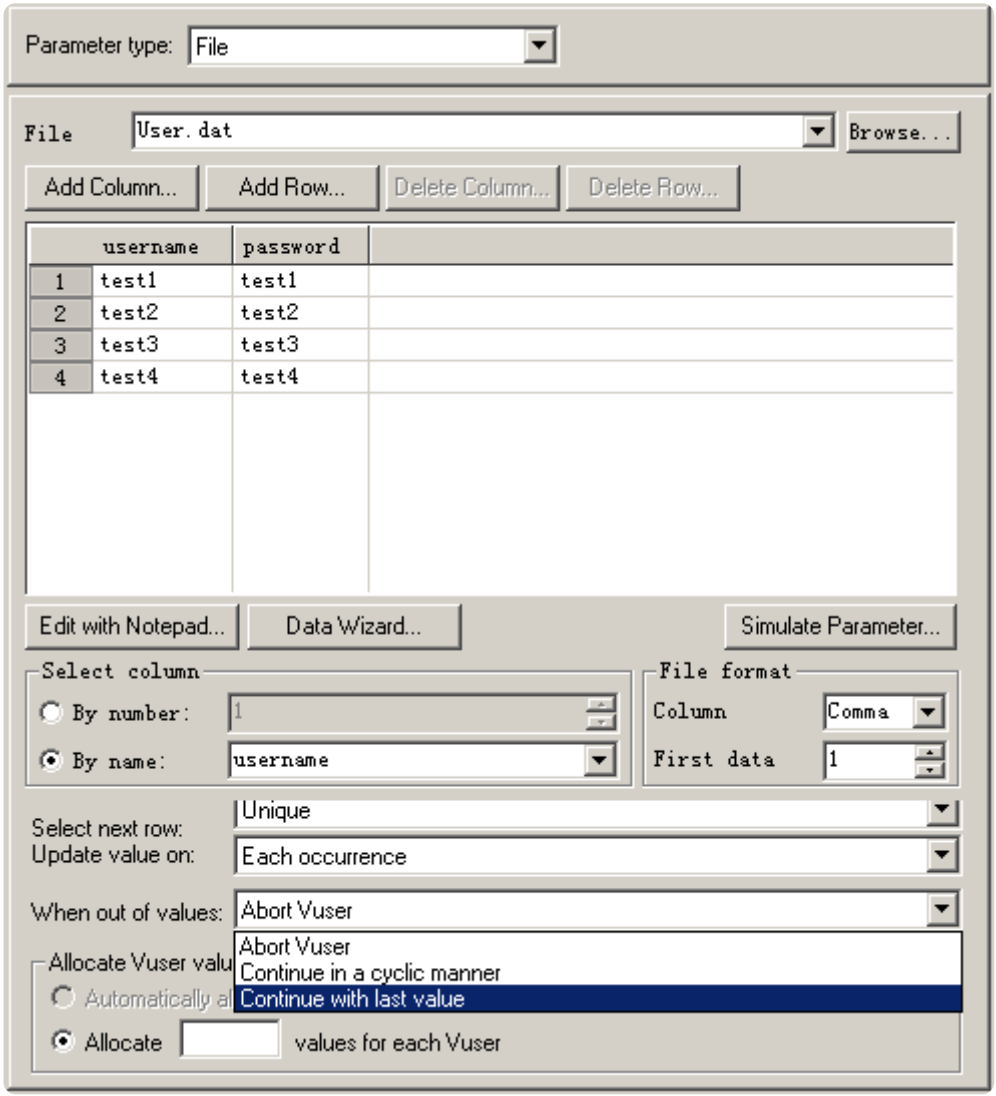
线程	迭代次数	取值
Thread Group 1 1-1	1	test00001
Thread Group 1 1-1	2	test00005
Thread Group 1 1-1	3	test00009
Thread Group 1 1-2	1	test00003
Thread Group 1 1-2	2	test00010
Thread Group 2 2-1	1	test00002
Thread Group 2 2-1	2	test00006
Thread Group 2 2-2	1	test00004
Thread Group 2 2-2	2	test00007
Thread Group 2 2-2	3	test00008

每次执行场景会有不同，不同点是线程组 1 有可能执行 6 次，而线程组 2 只执行 4 次；或者线程组 1 中的线程 2□执行次数比线程 1 执行次数多。但总体执行次数会是 10 次。

如果数据可以被线程平均分配，则每个线程的迭代次数会相同。如果数据不能被线程平均分配，则每个线程的迭代次数不会相同，但相差不会大。

参数化配置在 LoadRunner 中的使用说明

在 LoadRunner 中参数配置页面如下：



它的取值组合如下所示：

序号	取值方式	更新方式	值不足之后的处理方式
1	Sequential	Each Iteration	
2	Sequential	Each occurrence	
3	Sequential	Once	
4	Random	Each Iteration	
5	Random	Each occurrence	
6	Random	Once	
7	Unique	Each Iteration	Automatically allocate block size
8	Unique	Each Iteration	Allocate X values for each Vuser
9	Unique	Each occurrence	Allocate X values for each Vuser
10	Unique	Once	

以上的组合中，组合 7 对应着上文中 JMeter 真实场景中每次迭代取不同数据的组合，即 JMeter 中的参数组合 4。

总结

通过今天的内容，我们对性能测试中的参数化做了一次解析，在执行性能测试时，我们需要根据实际的业务场景选择不同的数据量和参数设置组合。

不同的压力工具在参数化的实现逻辑上也会不同，但是参数化必须依赖业务逻辑，而不是工具中能做到什么功能。所以在参数化之前，我们必须分析真实业务逻辑中如何使用数据，再在工具中选择相对应的组合参数的方式去实现。

这里我总结一下性能工作中参数化的逻辑，希望对你有所启发。

1. 分析业务场景；
2. 罗列出□需要参数化的数据及相对应的关系；
3. 将参数化数据从数据库中取出或设计对应的生成规则；
4. 合理地将参数化数据保存在不同的文件中；
5. 在压力工具中设置相应的参数组合关系，以便实现□模拟真实场景。

通过以上步骤，我们就可以合理的参数化数据，模拟出真实场景。

思考题

你可以思考一下下面几个问题：

1. 为什么参数化数据要符合生产环境的数据分布？
2. 为什么参数化数据要关注组合逻辑关系，而不是随意设置组合？

欢迎你在评论区写下你的思考，也欢迎把这篇文章分享给你的朋友或者同事，一起交流一下。

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇　09 | □关联和断言：一动一静，核心都是在取数据

下一篇　11 | 性能脚本：用案例和图示帮你理解HTTP协议

精选留言 (4)

写留言

律飛
2020-01-07

JMeter 的 CSV Data Set Config 功能用来从文件中读取数据行，并将它们拆分后存储到变量中。个人理解，Recycle on EOF的优先级高于Stop thread on EOF，也就是说，需要先判断Recycle on EOF，如果是Flase，直接在文件结束时就停止了线程，根本不考虑Stop thread on EOF参数值；如果是True，就要根据Stop thread on EOF参数值来确定线程是否停止运行。在明白组合逻辑关系后，可以更高效的设置参数、更准确的达到测试目...
展开

作者回复: 我觉得你写的比我写的好。哈。

1　1

小老鼠

2020-01-10

1，EOF处理不同对性能测试有什么影响？2，参数化用DB来获取，对性能测试结果的有无影响。

作者回复: 我觉得你思考问题的角度和我要说的话题都不在一个层面上。
1. EOF多明显对测试有影响，那使用数据的逻辑，我都列的那么清晰了，已经说明了影响了。
2. 参数化用不用DB来做，只会影响压力工具的处理过程。我反复强调了，对什么样的工具来做压力，对性能差的好的性能工具，都无所谓。像jmeter就不是性能高的工具。但是现在还是有很多人在用，对服务器的性能测试结果肯定是没什么影响的，服务器的处理能力是服务器的，是另一个角度。

晴空
2020-01-08

1、为什么参数化数据要符合生产环境的数据分布?
尽可能少的数据应用覆盖到尽可能多的测试场景中。
2、为什么参数化数据要关注组合逻辑关系，而不是随意设置组合?
关注组合逻辑关系，可以理解为了解数据生成规则，换句话说也就是有效的数据是有哪些实际业务场景产生的。...
展开

作者回复: 理解的很对。

善行通
2020-01-06

感谢老师总结;
1、罗列出需要参数化的数据及相对应的关系;
2、将参数化数据从数据库中取出或设计对应的生成规则;
3、合理地将参数化数据保存在不同的文件中;
4、在压力工具中设置相应的参数组合关系，以便实模拟真实场景...
展开

作者回复: 有收获我就值得了。